Delhi Technological University

PROJECT REPORT

Digital Electronics(DE)

Computer Program Counter Module

Aryan(2K19/SE/019)

Submitted to : **Dr. Deepika Sipal**

Table of contents

About the Topic

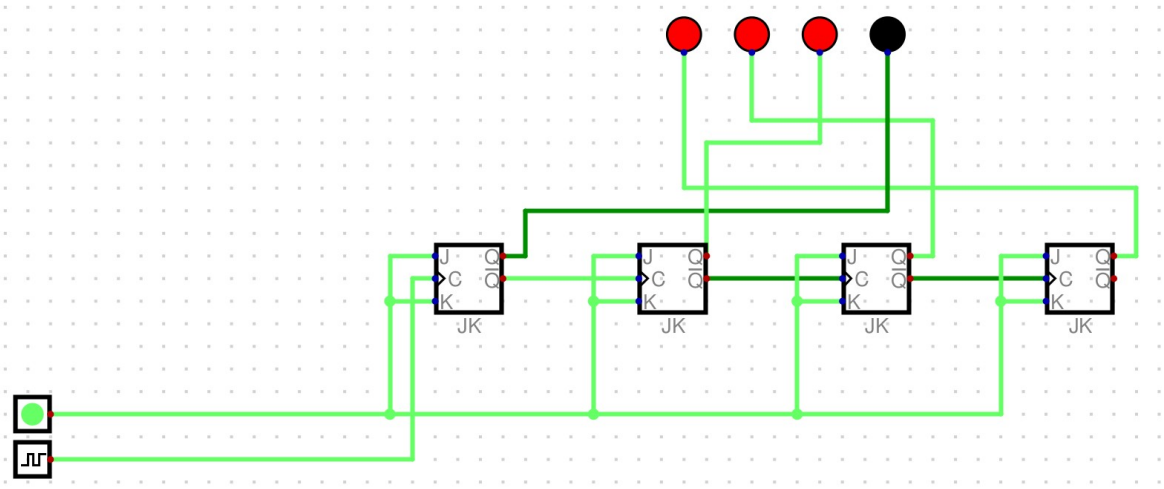
We made a software simulation of **Computer Program Counter Module**
A program counter is a register in a computer processor that contains the address(location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its stored value by 1 and it points to next instruction in the sequence.

Why this topic?

When we came to know about binary counter, it really showed how capable the digital circuits can be in contrast to the analog circuits.

As we learned more and more about the working of different kinds of counters we gained more and more insight, we were curious to test and make one ourselves so we made it in a simulating software and saw how it ran. We then realized that a similar process is occurs every nanosecond in the electronic devices we are currently using, this made us curious of the working of a program counter module in computers. We later did some research and found the working of a basic program counter quite easy to comprehend and implement, thus we decided to make a Program counter for a hypothetical computer using a variety of integrated circuits and other digital components.

Also one of the things that pushed us were our digital electronics practical labs as we always saw the practical implementation of the concepts we studied in our theory classes.

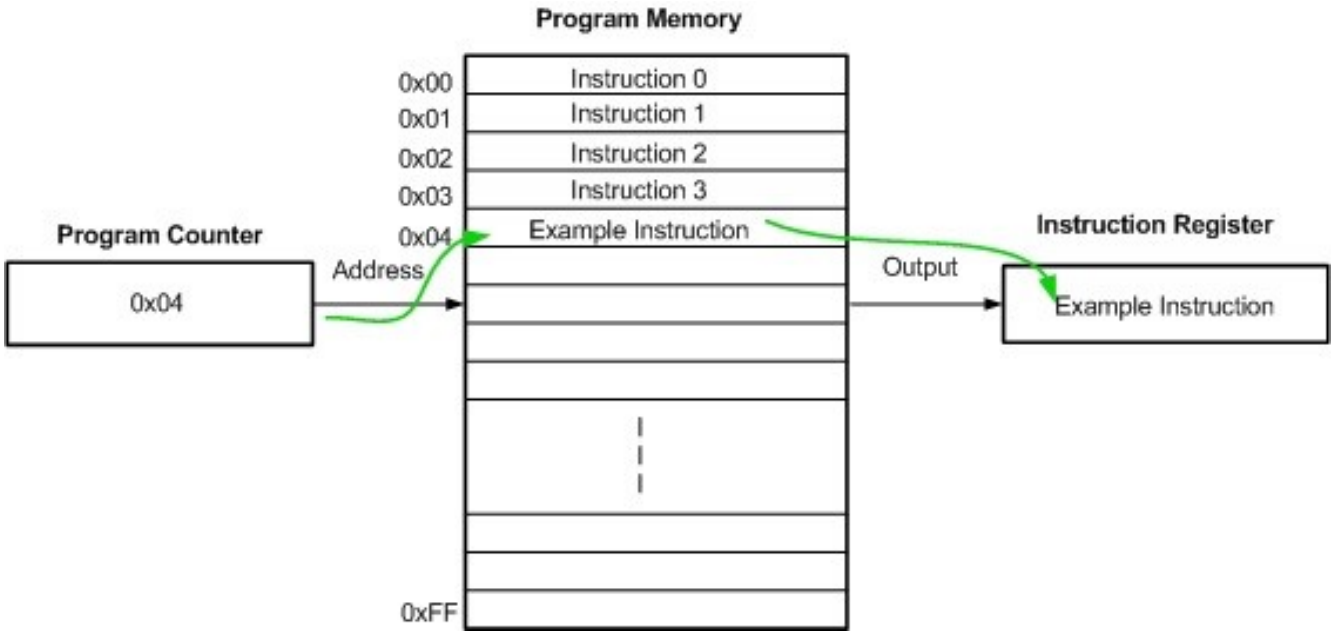


Theory

It is advised to watch the video of the simulation for a better idea

A counter by definition is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal. Counters are used in digital electronics for counting purpose, they can count specific event happening in the circuit. For example, in UP counter a counter increases count for every rising edge of clock.

The purpose of a program counter module in a computer is to hold the address of the next instruction that is to be executed. This control where the computer starts executing and where it goes next after that instruction is complete.



Thus a program counter need to to the following things:

- while a computer operates, it must know, what is the next instruction and where it has to go to fetch that instruction.
- Tell computer the next instruction in the queue.
- We must also be able to load data into the program counter.
- jump from a particular instruction(instead of executing sequentially), this jumping and pausing of program counter will also be implemented in the circuit.
- We must also be able to stop it at any time(at a clock pulse) for an instruction to completely execute.
- Communicate with the bus to communicate with other components of the computer.
- There must be able to control the output of the program counter i.e it must be able to get connected(putting all its data on the bus) or disconnected to the bus.

We initially tried implementing this circuitry using multiple JK flip flops of D flip flops but this turned out to very hard, so we researched a bit and then decided to stick with some standard integrated chips as:

- Using and building from scratch some standard repeatedly used components will be very difficult and tough to manage.
- Debugging would become unusually hard as there will be a mess of electronic components and wires.
- We are already aware of the conceptual working of these circuit so making each one from scratch would be redundant.

We used the following ICs in our project

- 74LS245 : This chip takes 8 input bits and either outputs them or disconnects them from the rest of the circuitry thus satisfying our need of disconnecting the program counter from the bus at will.
- 74LS161 X 2 : A single type of this chip acts as a four bit synchronous binary counter that can Load to the counter, Pause the counter and clear the counter synchronously with the clock pulse supplied.
- Two standard EEPROM chips for programming the computer.
- One standard ROM chips.
- A register whose usage will be explained ahead.

We later decided that we will be using this program counter to display a counting of single digit hex alphabets(0 to f) in order to demonstrate how our program counter is able to perform all of its functions.
We had no other choice other than emulating that a program is running on our simulated hypothetical computer as were did not have enough time and competency to develop a whole computer.

Thus we used two ROMs :

- One to emulate that a hypothetical program is running on our computer by giving it appropriate control signals(coded in machine language) at every clock pulse.
- The other two EEPROMs were used for converting the binary number received to a binary signal that is able to display all the numbers from 0 to f in our seven segment display.

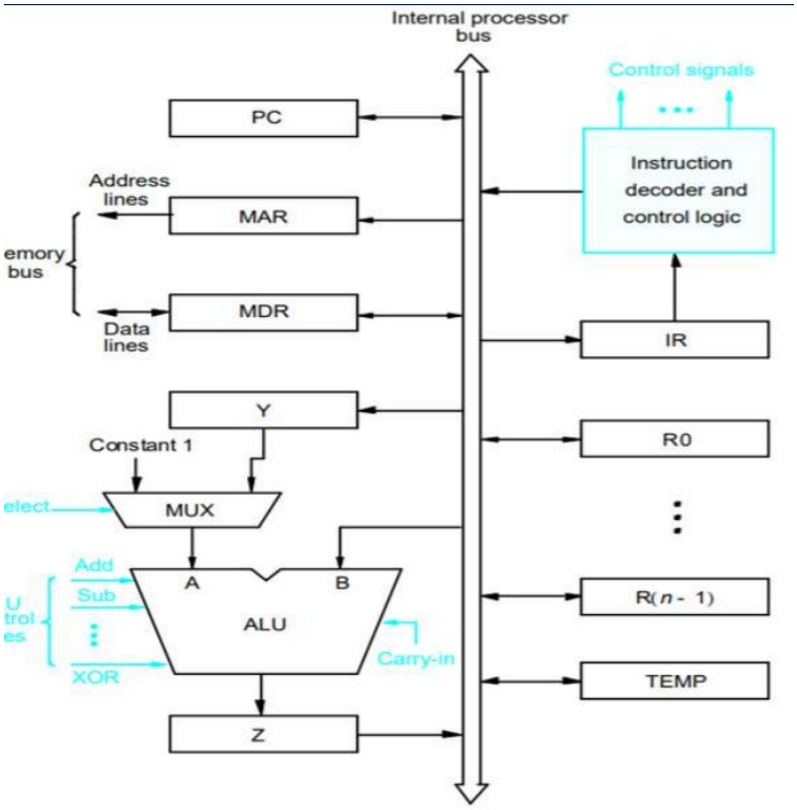
Systematic approach(in form of a gantt chart)

Included in the excel file attached with the document.

Working

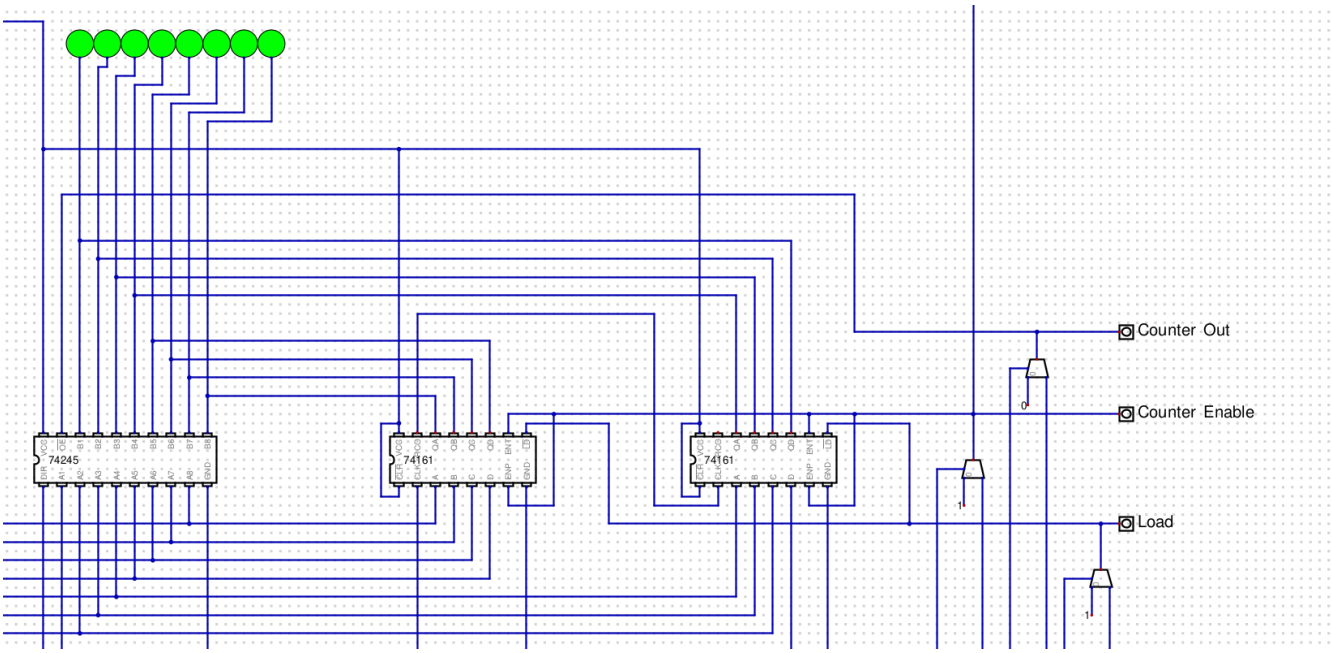
Design

Our program counter design is very simple by design and is expected to connect to a computer in the following way



The above image shows the required program counter in a single bus computer,
We will only be implementing the complete Program Counter(PC) unit and clock and communication hardware to communicate with the bus.

Our program counter design:

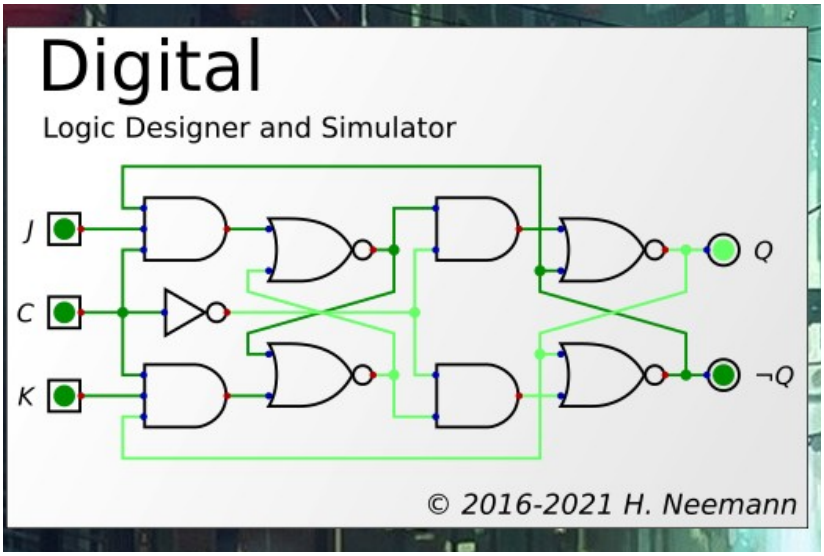


as the complete circuit is quite big to not fit in this document in a readable manner, **so I have uploaded the complete high resolution image of the circuit with this document**.

Implementation

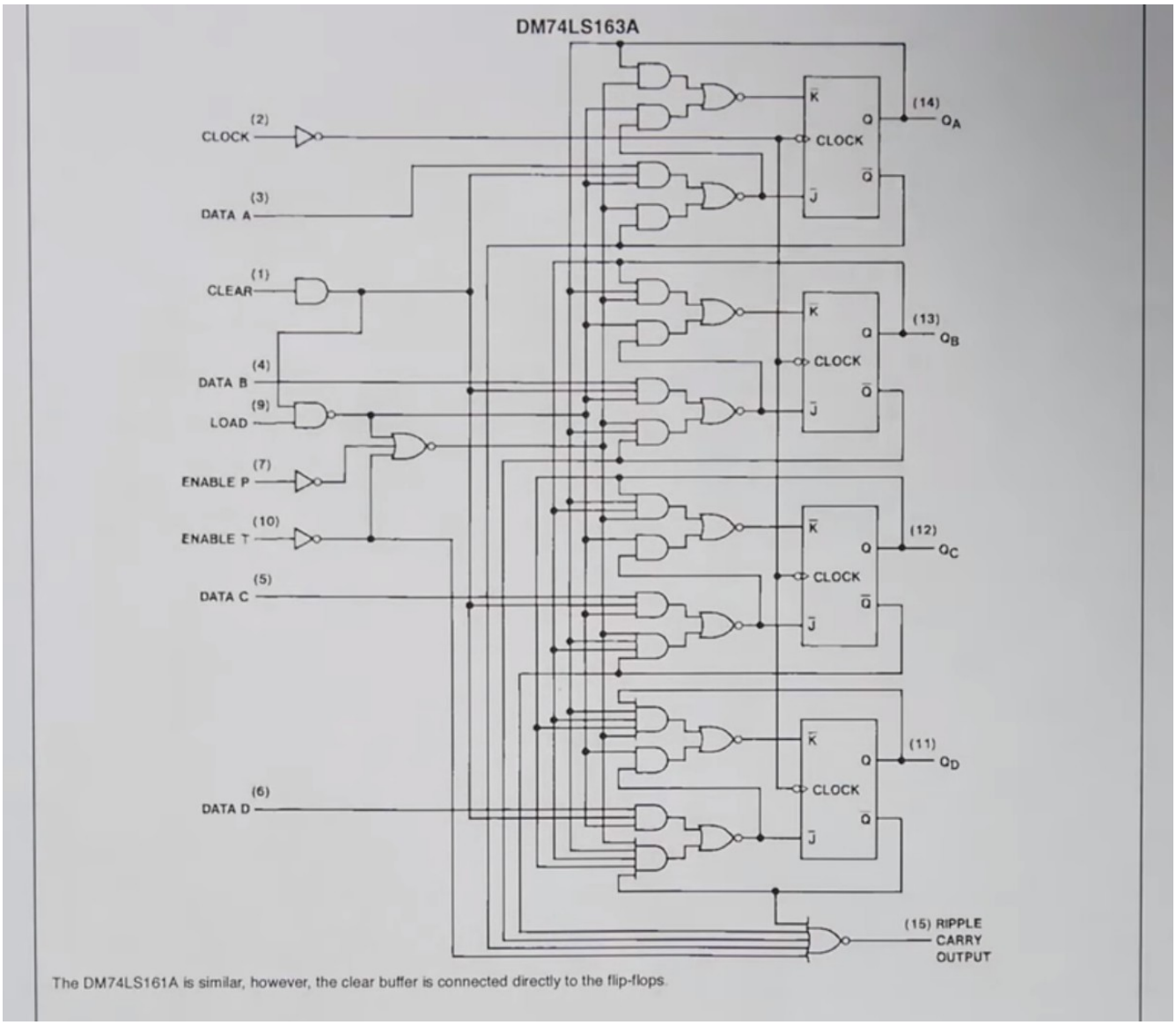
After a lot of research trials and testing we decided to use the following simulation software :

Digital logic designer and simulator by H. Neemann
It is an open source software with GNU GPL, available on github :
https://github.com/hneemann/Digital



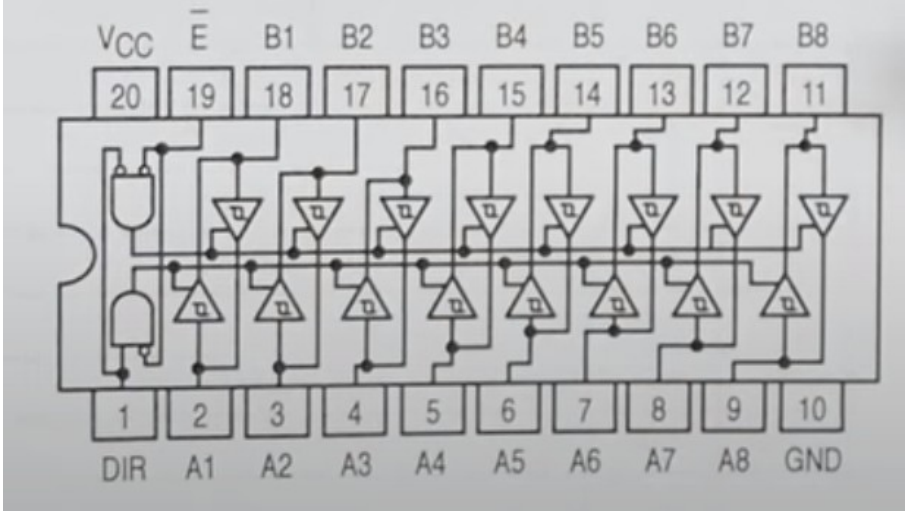
As we explained the brief working of a program counter, let us now describe how we were able to make it work.

As can be seen by the image below :



This is the picture of the internal circuitry of the 74LS161X chip, we can see the 4 JK flip flops in the chip and there is also some additional logic about which we will not go in detail but this circuit is able to provide us enable, load and clear(not used) functionality.

Similarly, the internal circuitry of the DM74LS245A chip exposes the 8 tri-state selectors it has which can either output the input or disconnect the input entirely from the the output lines.



Then, we made a simple circuit proposed by Ben eater (website linked in references). We were able to do this in a few days as it wasn’t much complex, but as there was no way of testing the circuit, we tried to implement the following in order to test it.

Testing

To test the circuit, we needed a compatible computer executing programs, this certainly was not possible as simulating a computer with all of its components would be unreasonably difficult, so we decided to emulate this by giving it appropriate control signals(coded in machine language) at every clock pulse, whenever needed.

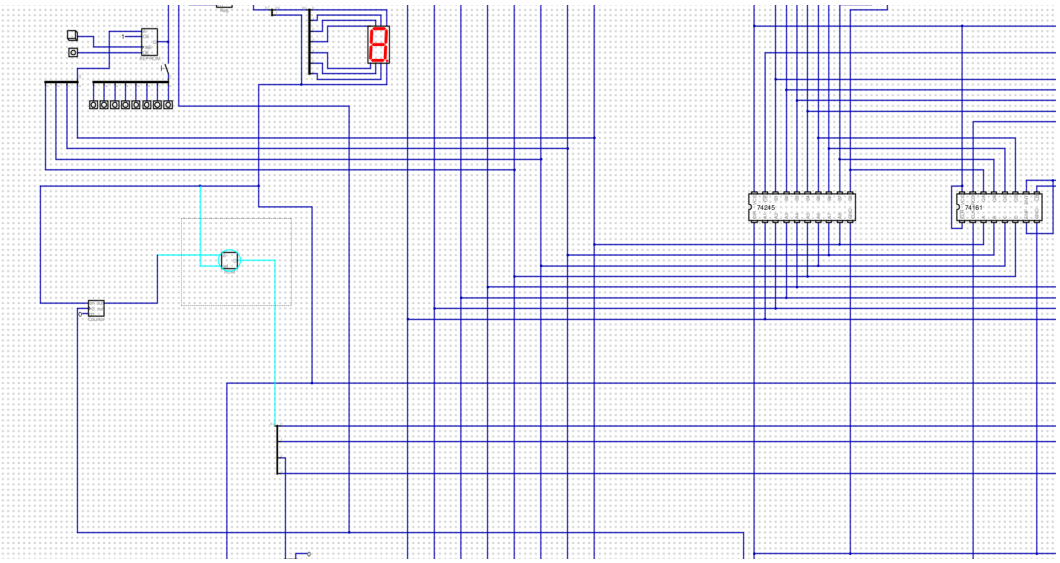
We planned and implemented the following :

- We made a seven segment display and programmed it to display the hex equivalent to show the program counter out functionality.
- We programmed a single bit in the EEPROM of the seven segment to initiate a simple basic program, the use of this bit will become clear as we read.
- It should be noted that it took over a month to test the circuit and get our supposed program working, so as all that testing and debugging will make this report very long we decided to mention all the failures in brief.
- We took this bit and connected it to the control signal inputs to emulate a jump scenario but we encountered a bunch of problems:
  1. The program went into a loop and did not produce the desired output.
  2. As the control signals were hard wired, the chance of short circuit increased drastically.
  3. It turned out that it was not possible for us to perform the whole load sequence in one or two clock cycles as there were many steps included.
  4. Also as the Seven segment display was displaying the output directly from the bus all the time and that one bit we mentioned above was also dependent on the output of the bus thus, as soon as the the address on the bus changes that changes the output at that control bit thus, the whole circuit went haywire.

- Then to mitigate these errors we did the following things :
  1. We added a register that stores the input from the bus(depending on whether it is enabled or not) and outputs the stored data to the seven segment display which after getting decoded is displayed in hex decimal format.
  2. We added many multiplexers to give control signal only when we desire, they act kind of like tri-state buffers except there was more than one input, this was done in order to avoid short circuits.
  3. We added another ROM(third) to provide appropriate control signals to the computer and perform the task(jumping to another address) in multiple clock cycle(8 clock pulses in our simulation).
- We were then after some trial and error were able to make the circuit run in the intended way, which can be seen in the simulation video we have uploaded with this document.

Now to further describe the video by mentioning the code for the control signals in binary : these were programmed into this ROM

0	0b1000
1	0b1010
2	0b1110
3	0b0110
4	0b1110
5	0b1010
6	0b1000
7	0b1001
8	0b0000
9	0b0000
0xA	0b0000
0xB	0b0000
0xC	0b0000
0xD	0b0000
0xE	0b0000
0xF	0b0000



DATE
PAGE

We follows the following to generate control signals for our jump instruction

1. Disable Counter
2. Disable output of Program Counter
3. Give address to be loaded to the bus
4. Make load Pin low
5. Make load Pin back high
6. Remove address from the bus
7. enable output of Program Counter
8. enable the program counter

here, we consider the 4 bits as following Control Signals i-e

0 → enable Counter Pin
1 → enable out Pin (active low)
2 → load address to bus
3 → PC load Pin (active low)

thus we get the instruction as:

3	2	1	0
1	0	0	0
1	0	1	0
1	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
1	0	0	0
1	0	0	1

it should be noted that the Default state of Control signals before instruction starts executing is 1001 which is also the last state

The above page shows the calculations performed for the jump instruction to execute successfully, this also makes the control signals programmed in the ROM more clear.

Applications

The application of the program counter circuit is endless, but currently it is being used in almost any smart digital device as any computer irrespective of its scale cannot function without a program counter module.

There are many diverse applications of these kinds of circuit some of which are not even thought of or invented yet.

We basically used the program counter to run a supposed program that counts from 0 to f and may even jump if we program it to make it do that.

Conclusion

The main purpose of our program was to make us learn something which it certainly succeeded in doing.

We were thus able to make a program counter with all its working functionalities i.e program counter enable , program counter load, program counter output control, program counter jump and of course program counter count(enable).

Thus to conclude it was very exciting making this project, we were so amazed when it finally worked the way we expect it to, we also forked it and tried some new things on the fork which also produced promising results.

References

1. https://www.alldatasheet.com/view.jsp

2. https://eater.net/8bit

3. https://www.youtube.com/watch?v=ztkMRln_jDA&list=PLmXKhU9FNesSfX1PVt4VGm-wbIKfemUWK

4. Digital Design by M. Morris Mano

5. Notes provided by our professor Dr. Deepika Sipal